UIO Sequence Based Checking Sequences for Distributed Test Architectures

R. M. Hierons[1] and H. Ural[2]

1 Department of Information Systems and Computing, Brunel University, Middlesex, UB8 3PH, United Kingdom
2 School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, K1N 6N5, Canada

Abstract

This study addresses the construction of a preset checking sequence that will not pose controllability (synchronization) and observability (undetectable output shift) problems when applied in a distributed test architectures that utilize remote testers. The controllability problem manifests itself when a tester is required to send the current input and because it did not send the previous input nor did it receive the previous output it cannot determine when to send the input. The observability problem manifests itself when a tester is expecting an output in response to either the previous input or the current input and because it is not the one to send the current input,-it cannot determine when to start and stop waiting for the output. Based on UIO sequences, a checking sequence construction method is proposed to yield a sequence that is free from controllability and observability problems.

Keywords

Distributed testing, Controllability, Observability, Test coordination, UIO sequences, Checking sequences

1 INTRODUCTION

Determining, under certain assumptions, whether a given "black box" implementation *N* of a Finite State Machine (*FSM*) *M* is functioning correctly is referred to as a *fault detection* (*checking*) *experiment*. Foundations of fault detection experiments can be found in the sequential circuit testing literature [GI 62, HE 64]. This experiment is based on an input sequence called a *checking sequence* constructed from a given deterministic and minimal FSM *M* with a designated initial state that determines whether a given FSM *N* is a correct implementation of *M*. The construction of a checking sequence must deal with the "black box" nature of a given implementation *N* of *M* which allows only limited controllability and observability of *N*. The limited controllability refers to not being able to directly transfer *N* to a designated state and the limited observability refers to not being able to directly recognize the current state of *N*. In order to overcome the restrictions imposed by the limited controllability and observability, some special input sequences must be utilized in the construction of a checking sequence such that the output sequences produced by *N* in response to these input sequences provide sufficient information to deduce that every state transition of *M* is implemented correctly by *N*.

In order to verify the state transition from state *a* to *b* under input *x*, 1) before the application of *x*, *N* must be transferred to the state recognized as *a*, 2) the output produced by *N* in response to the application of *x* must be as specified in *M*, and 3) the state reached by *N* after the application of *x* must be recognized as *b*. Hence, a crucial part of testing the correct implementation of each transition is recognizing the starting and terminating states of the transition. The recognition of a state of an FSM *M* can be achieved by a distinguishing sequence [HE 64], a characterization set [HE 64] or a unique input-output (UIO) sequence [SD 88]. It is known that UIO sequences may not exist for every state of every minimal FSM [SD 88] and determining the existence of a UIO sequence for a state of an FSM is PSPACE-complete [LY 94]. Nevertheless, based on UIO sequences, various methods have been proposed in the literature to test FSMs [SD 88, AA 88, CV 89, DS 90, HI 97]. Some of these methods have been used to construct fault detection experiments to test the conformance of various protocol implementations to their specifications given as FSMs [CV 89].

Testing an implementation *N* of an FSM *M* can be carried out as a fault detection experiment in some specific test architectures. One such architecture is the distributed test architecture shown in Figure 1 [II 95] where the lower interface and the upper interface of the implementation *N* may be controlled and observed indirectly by the lower tester (*L*) and directly by the upper tester (*U*), respectively.

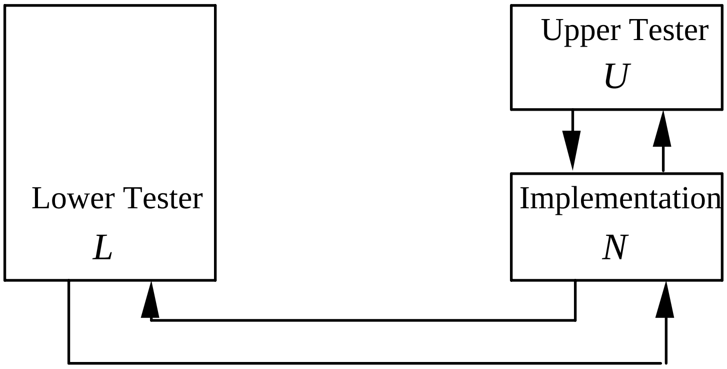

Figure 1 A Distributed Test Architecture

In this architecture, *U* and *L* are two remote testers that are required to coordinate the application of a preset checking sequence through their interactions with *N*. However, this requirement may lead to controllability and observability problems, in addition to those that stem from the black box nature of *N*. The *controllability (synchronization) problem* manifests itself when *L* (or *U*) is expected to send an input to *N* after *N* responds to an input from *U* (or *L*) with an output to *U* (or *L*), but *L* (or *U*) is unable to determine whether *N* sent that output. It is therefore important to construct a synchronizable checking sequence that causes no controllability problem during its application in the distributed test architecture.

During the application of even a synchronizable checking sequence in a distributed test architecture, the observability problem manifests itself when *L* (or *U*) is expected to receive an output from *N* in response to either the previous input or the current input and because *L* (or *U*) is not the one to send the current input, *L* (or *U*) is unable to determine when to start and stop waiting. Such observability problems hamper the detectability of *output shift faults* in *N* i.e., an output associated with the current input is generated by *N* in response to either the previous input or the next input. To ensure the detectability of output shift faults in *N* the checking sequence needs to be augmented by additional input subsequences.

Based on the work presented in [GU 95], this paper proposes a method for constructing a checking sequence that does not pose controllability and observability problems during its application in a distributed test architecture. Earlier work on the controllability problem [SB 84, BU 91, UW 93, CU 95, TY 98] and that of on the observability problem [LB 94, YT 98, CR 99] consider the construction of a test sequence rather than a checking sequence. It is well known that the complete fault coverage of a checking sequence cannot be directly achieved by a test sequence where transition verification is not necessarily based on state verification.

The rest of the paper is organized as follows: Related terminology is reviewed in Section 2. In Section 3, the proposed method is presented and a proof for the resulting sequence to be a checking sequence is given. An illustrative example of the application of the proposed method is provided in Section 4. In Section 5, some minimization techniques are proposed and concluding remarks are given.

2 PRELIMINARIES

2.1 FSM and its Graphical Representation

A *finite state machine*, *FSM*, is a quintuple $M = (S, X, Y, \delta, \lambda)$, where S, X, Y are finite sets of states, inputs, outputs, δ is a state transition function that maps $S \times X$ to S, and λ is an output function that maps $S \times X$ to Y. State $s_1 \in S$ is designated as the *initial state* of M, and $|S|$ is n. Each element $y \in Y$ is in the form of one of the pairs (-, -), $(o^U, -)$, $(-, o^L)$, and (o^U, o^L) where - denotes *null* output; and the first and second element in each pair is an output to U and L, respectively. Functions δ and λ are extended to a sequence of inputs in the standard manner. An FSM M is said to be *minimal* if $\forall$ $(s_i, s_j) \in S^2$, $i \neq j$, $\exists$ $I \in X^*$, such that $\lambda(s_i, I) \neq \lambda(s_j, I)$. An FSM M is said to be *completely specified* if $\forall x \in X$, $\forall$ $s \in S$, $\exists$ $(s, \delta(s, x); x/\lambda(s, x))$ in M. An FSM M can be represented by a directed graph (digraph) $G = (V, E)$ where a set of vertices $V = \{1, 2,..., n\}$ represents the set S of states of M, and a set of directed edges $E \subseteq \{e \mid e = (v_i, v_j; x/y), v_i, v_j \in V\}$ represents the set of transitions of M. An edge $e = (v_i, v_j; x/y) \in E$ represents a specified transition $t = (s_i, s_j; x/y)$ of M from state s_i to state s_j with input $x \in X$ and output $y \in Y$. Vertices v_i and v_j which represent respectively state s_i and s_j are called the *head* and the *tail* of e, denoted *head*(e) and *tail*(e). The input/output pair (or *io-pair*, in short) x/y is called the *label* of the edge e (or transition t), denoted *label*(e) (or *label*(t)).

A *path* $P = (n_1, n_2; x_1/y_1)(n_2, n_3; x_2/y_2) \ldots (n_{k-1}, n_k; x_{k-1}/y_{k-1})$, $k > 1$, in a digraph $G = (V, E)$ is a finite sequence of adjacent (not necessarily distinct) edges in G, where n_1 and n_k, are called the *head* and the *tail* of P, denoted *head*(P) and *tail*(P)**,** respectively, and $(x_1/y_1)(x_2/y_2) \ldots (x_{k-1}/y_{k-1})$, is called the *label* of P, denoted *label*(P). For convenience, a path $P = (n_1, n_2; x_1/y_1)(n_2, n_3; x_2/y_2) \ldots (n_{k-1}, n_k; x_{k-1}/y_{k-1})$ will be represented by $(n_1, n_k; I/O)$ where *label*(P) = I/O is the input-output (or in short, IO-) sequence $(x_1/y_1)(x_2/y_2) \ldots (x_{k-1}/y_{k-1})$, $I = x_1x_2 \ldots x_{k-1}$ is the *input portion* of I/O, $O = y_1y_2 \ldots y_{k-1}$ is the *output portion* of I/O, respectively. A sequence $(i_1i_2 \ldots i_k)$ is a *subsequence* of $(x_1x_2 \ldots x_m)$ if there exists a Δ, $0 \leq \Delta \leq m\text{–}k$, such that for all j, $1 \leq j \leq k$, $i_j = x_{j+\Delta}$. A sequence $(i_1i_2 \ldots i_k)$ is a *prefix* of $(x_1x_2 \ldots x_m)$ if $\forall$ j, $1 \leq j \leq k$, $i_j = x_j$.

A digraph $G = (V, E)$ is *strongly connected*, if for every pair of vertices v_j and v_k, there exists a path from v_j to v_k. An FSM M has the *reset feature* if there is an input r such that $\delta(s_i, r) = s_1$ and $\lambda(s_i, r)$ = (-, -), for every state s_i of M. A *transfer sequence T* of an FSM M from state s_i to state s_j is the input portion of the label of a path from s_i to s_j.

Let $\Phi(M)$ be the set of FSMs each of which has at most n states and the same input set as M. Let N be in $\Phi(M)$. N *conforms* to M if there is a one-to-one and onto function f on the state sets of M and N such that for any transition $(i, j; x/y)$ of M, $(f(i), f(j); x/y)$ is a transition of N. A *checking sequence* of M is an input sequence starting at a specific state of M that distinguishes M from any FSM of $\Phi(M)$ which does not conform to M.

2.2 Controllability (Synchronization) Problem

Let each transition t of an FSM M have one of the following labels, *label*(t) $\in \{i^L/\text{-}, i^U/\text{-}, i^L/o^L, i^L/o^U, i^U/o^L, i^U/o^U, i^L/o^{U,L}, i^U/o^{U,L}\}$ where i^L (i^U) is an input from $L(U)$, $o^L(o^U)$ is an output to

$L(U)$ and $o^{U,L}$ is an output to L and U. Then, considering two consecutive transitions of M, one of the testers, say L (or U), faces a *synchronization problem* if L (or U) did not take part in the first transition and if the second transition requires that it sends an input to M [SB 84]. For example, a synchronization problem will occur if t_1 is followed by t_2 and $label(t_1) = i^L/o^L$ whereas $label(t_2) \in \{i^U/\text{-}\ , i^U/o^L, i^U/o^U, i^U/o^{U,L}\}$.

Two adjacent transitions t and t' of M form a *synchronizable pair of transitions* if t' can follow t without generating a synchronization problem. For example, a transition with label i^L/o^U forms a synchronizable pair of transitions when followed by any other transition of M. For a transition t of an FSM, each transition t' that forms a synchronizable pair of transitions with t is called an *eligible successor* of t. Using the notion of eligible successor, one can determine, in polynomial time, whether there is a non-empty subset of edges specified as eligible successors of e for each edge $e \in E$ in a given digraph $G = (V, E)$ [BU 91]. A transition sequence of an FSM (or a path of the corresponding digraph $G = (V, E)$) is *synchronizable* if for every two consecutive transitions in the sequence (or edges in the path), the second transition is an eligible successor of the first one. An input-output (input) sequence is *synchronizable* if it is (the input portion of) the label of a synchronizable path.

2.3 Observability Problem

An *observability problem* exists when a tester θ (= U or L) is expecting an output in response to either the previous input or the current input and because it is not the one to send the current input, it cannot determine when to start and stop waiting for the output. The observability problem manifests itself as an undetectable output shift fault. Formally, given an FSM M and a test sequence x_1/y_1, x_2/y_2, …, x_m/y_m of M, where $x_i \in X$ and $y_i \in Y$, $1 \le i \le m$, an *output shift fault* in an implementation N of M occurs when, in the labels x_j/y_j and x_{j+1}/y_{j+1} of any two consecutive transitions, $1 \le j \le m\text{-}1$, there exists an output o^θ (= o^U or o^L) that is

- either in y_j of M that occurs in y_{j+1} of N and not in y_j of N (*forward* output shift fault);
- or in y_{j+1} of M that occurs in y_j of N and not in y_{j+1} of N (*backward* output shift fault).

In general, if there is an output shift fault related to an output o^θ in any two consecutive transitions whose labels are x_j/y_j and x_{j+1}/y_{j+1}, this fault will not be detected by tester θ that satisfies the following condition

(o^θ is in y_j of M XOR o^θ is in y_{j+1} of M) AND input $x_{j+1}$is not sent by tester θ.

In this case, we say that the tester θ is *involved* in the shift.

3 THE PROPOSED METHOD

Let $M = (S, X, Y, \delta, \lambda)$ hereafter stand for a minimal FSM which is represented by a strongly connected digraph $G = (V, E)$ where there is a non-empty subset of outgoing edges of vertex v_j specified as eligible successors of e ($tail(e)=v_j$), for each edge $e \in E$. Let $|S|$ be n and $s_1 \in S$ be the initial state of M. The construction of a synchronizable checking sequence of M is based on the construction of a digraph $G' = (V', E')$ such that there is a one-to-one mapping from the edges in G to the edges in G', and that every pair of edges (corresponding to a pair of adjacent edges in G) covered by some path in G' is a synchronizable pair of transitions in M. Thus, finding a synchronizable checking sequence on G will be reduced to finding a checking sequence on G'. After the checking sequence is formed, it is examined for potentially undetectable output shift faults. Each potentially undetectable output shift fault is eliminated in the checking sequence without creating any synchronization problem by adding some subsequences to the checking sequence. The resulting checking sequence can then be applied in a distributed test architecture without creating controllability or observability problems.

3.1 Construction of the Digraph *G'*

For each vertex v in $G = (V, E)$, define:
$Depart^{U}[v] = \{e \in E : head(e) = v$ and $label(e) \in \{i^{U}/\text{-}, i^{U}/o^{U}, i^{U}/o^{L}, i^{U}/o^{U,L}\}\}$
$Depart^{L}[v] = \{e \in E : head(e) = v$ and $label(e) \in \{i^{L}/\text{-}, i^{L}/o^{L}, i^{L}/o^{U}, i^{L}/o^{U,L}\}\}$
$Arrive^{U}[v] = \{e \in E : tail(e) = v$ and $label(e) \in \{i^{U}/\text{-}, i^{U}/o^{U}\}\}$
$Arrive^{L}[v] = \{e \in E : tail(e) = v$ and $label(e) \in \{i^{L}/\text{-}, i^{L}/o^{L}\}\}$
$Arrive^{U,L}[v] = \{e \in E : tail(e) = v$ and $label(e) \in \{i^{L}/o^{U}, i^{U}/o^{L}, i^{L}/o^{U,L}, i^{U}/o^{U,L}\}\}$.

Note that the edges in $Depart^{U}[v]$ are eligible successors of edges in $Arrive^{U}[v]$ and $Arrive^{U,L}[v]$. The edges in $Depart^{L}[v]$ are eligible successors of edges in $Arrive^{L}[v]$ and $Arrive^{U,L}[v]$. Furthermore, edges in $Depart^{U}[v]$ are the only eligible successors of edges in $Arrive^{U}[v]$, and edges in $Depart^{L}[v]$ are the only eligible successors of edges in $Arrive^{L}[v]$.

During the construction of $G' = (V', E')$ from $G = (V, E)$, for each vertex $v \in V$, 1, 2 or 3 vertices are created in G', depending on the labels of edges arriving and departing v, so that if an edge arrives at a vertex in V', it can take as its eligible successor any of the edges departing at this vertex. The procedure of constructing the digraph $G' = (V', E')$ from a strongly connected digraph $G = (V, E)$ such that $V' = V^{U} \cup V^{L} \cup V^{U,L}$ and $E' = E_c \cup F$ is as follows:

(1) For each vertex v in V,
If $Depart^{U}[v] \neq \emptyset$, then a vertex v^{U} is created in V^{U}.
If $Depart^{L}[v] \neq \emptyset$, then a vertex v^{L} is created in V^{L}.

(2) For each edge $(w, v; x/y) \in Arrive^{U}[v]$ (this implies that $(w, v; x/y) \in Depart^{U}[w]$), a directed edge from w^{U} to v^{U} is created in E_c. Similarly, for each edge $(w, v; x/y) \in Arrive^{L}[v]$ (this implies that $(w, v; x/y) \in Depart^{L}[w]$), a directed edge from w^{L} to v^{L} is created in E_c.

(3) For each edge $(w, v; x/y) \in Arrive^{U,L}[v]$, one of the following is performed:

a) In the case that vertex v^{U} exists but vertex v^{L} does not,
if $(w, v; x/y) \in Depart^{U}[w]$, then a directed edge from w^{U} to v^{U} is created in E_c,
else (i.e., $(w, v; x/y) \in Depart^{L}[w]$) a directed edge from w^{L} to v^{U} is created in E_c.

b) In the case that vertex v^{L} exists but vertex v^{U} does not,
if $(w, v; x/y) \in Depart^{U}[w]$, then a directed edge from w^{U} to v^{L} is created in E_c,
else (i.e., $(w, v; x/y) \in Depart^{L}[w]$) a directed edge from w^{L} to v^{L} is created in E_c.

c) In the case that both v^{U} and v^{L} exist, a vertex $v^{U,L}$ is created in $V^{U,L}$ (if it does not already exist) and two edges $(v^{U,L}, v^{U})$ and $(v^{U,L}, v^{L})$ with nil labels, denoted by -/(-, -) when needed, are constructed in F.
If $(w, v; x/y) \in Depart^{U}[w]$, then a directed edge $(w^{U}, v^{U,L}; x/y)$ is created in E_c,
else (i.e., $(w, v; x/y) \in Depart^{L}[w]$) a directed edge $(w^{L}, v^{U,L}; x/y)$ is created in E_c.

It follows from the construction of $G' = (V', E')$ that

- for each edge $e \in E$ of G there is exactly one corresponding edge $e' \in E_c$ of G'
- for each edge $e' \in E_c$ of G' there is exactly one corresponding edge $e \in E$ of G.

Given an edge $e \in E$ of G, let $f(e)$ denote the corresponding edge $e' \in E_c$ of G'. Then f can be extended to be applied to sequences of edges. Given path p' in G', let $g(p')$ denote p' with the edges from F removed. Then, a path p' in G' that starts at $v_1^{U,L}$ is said to *correspond* to a path p

of G that starts at v_1 if, and only if, $f(p)=g(p')$. If path p' in G' that starts at $v_1^{U,L}$ corresponds to path p of G that starts at v_1, then label(p) = label($g(p')$).
From the construction of $G' = (V', E')$, the following propositions are immediate:

Proposition 1: For each synchronizable path p in G that starts at v_1, there is a corresponding path p' in G' that starts at $v_1^{U,L}$.
Proposition 2: For each path p' in G' that starts at $v_1^{U,L}$, there is a corresponding synchronizable path p in G that starts at v_1.
Proposition 3: Every vertex $v \in V'$ of G' is reachable from $v_1^{U,L}$.

3.2 Construction of Synchronizable Checking Sequences

Checking sequences that will be formulated in this paper will utilize synchronizable UIO sequences. A *Unique input-output* (UIO) sequence for a state of an FSM is an IO-sequence that is not exhibited by any other state of the FSM [SD 88]. A *synchronizable unique input/output* (SUIO) sequence for a state s of an FSM is a synchronizable IO-sequence that forms a UIO for s. Although an SUIO sequence itself does not cause any synchronization problem, a synchronization problem can still arise if the transition corresponding to the first io-pair of the SUIO sequence is not an eligible successor of the transition which precedes the SUIO sequence in a given transition sequence. For example, when an SUIO sequence for a state s starts with an input sent by U, any incoming transition of state s with label i^L/o^L or i^L/-, will cause a synchronization problem if it precedes the SUIO sequence. To avoid a synchronization problem, for each state s, two SUIO sequences may be needed, denoted by SUIO^U(s) and SUIO^L(s) where SUIO^U(s) starts with an input sent by U and SUIO^L(s) starts with an input sent by L.

A general approach for the construction of synchronizable checking sequences based on UIO sequences is first to construct a *state cover* which is an input sequence used to verify that an implementation N of a given FSM M has all the states of M. This is followed by a *transition cover* which is an input sequence used to verify that N implements all transitions of M. In order to verify each state s of M in N, not only the input portion of the UIO(s) is applied to s, but also the input portions of UIOs of every other state is also applied to s so that the uniqueness of UIOs in N can also be established. If N passes the state cover then, the verification of each transition $t = (s_i, s_j; x/y)$ of M in N is attempted by bringing N to s_i, applying the input x, verifying that the observed output is y, and verifying that N reaches s_j by applying the input portion of UIO(s_j).

The proposed method for the construction of a synchronizable checking sequence from M is based on the following observations regarding an FSM from which we may generate a synchronizable checking sequence using SUIOs:

A) If for each state $s \in S$, *Arrive*$^U[s] = \emptyset$, then no state in S needs to have an SUIO^U.
   Further, if no state in S has an SUIO^U then for each state $s \in S$, *Arrive*$^U[s]$ must be $\emptyset$.
   A also holds for L.
B) If for at least one state $s \in S$ (represented by $v \in V$), *Arrive*$^U[s] \neq \emptyset$ then
   a) *Depart*$^U[s]$ must not be $\emptyset$
      (as edges in *Depart*$^U[s]$ are the only eligible successors of edges in *Arrive*$^U[s]$).
   b) s needs to have an SUIO^U, i.e., SUIO^U(s) with an input portion I_{vU}
      (as the verification of edges in *Arrive*$^U[s]$ requires an SUIO that starts with the label of an edge in *Depart*$^U[s]$).
   c) SUIO^U(s) must be used in the verification of s in N
      (as the verification of edges in *Arrive*$^U[s]$ requires the use of SUIO^U(s)).

d) for each state $s' \in S$,
   - $Depart^U[s']$ must not be $\emptyset$,
   - $Arrive^U[s']$ or $Arrive^{U,L}[s']$ must not be $\emptyset$,
   - I_{vU} must be the input portion of the label of a synchronizable transition sequence that starts at s', (as the verification of s in N requires the use of SUIO$^U(s)$ whose uniqueness in N must be established), and
   - At least one of the edges in $Arrive^U[s']$ or $Arrive^{U,L}[s']$ must form a synchronizable pair of transitions with the transition in $Depart^U[s']$ that maps to the first transition of the sequence induced by I_{vU} at s'.

   B also holds for L.

Before the proposed method is presented the following assumptions are made:

1) The implementation N of M implements the reset feature of M correctly. A reset transition has label r/(-, -) and the reset input r can be sent by any of the testers and can be followed by any input from any tester without causing any synchronization problem. N is deterministic, minimal, composed of at most n states, and complete.
2) For each edge $e \in E$, there is a synchronizable path that starts at the initial vertex v_1 and contains e.
3) For each state s, represented by vertex $v \in V$, if $Arrive^U[v]$ (resp. $Arrive^L[v]$) $\neq \emptyset$, then SUIO$^U(s)$ (resp. SUIO$^L(s)$) exists and if $Arrive^{U,L}[v] \neq \emptyset$, then at least one of SUIO$^U(s)$ or SUIO$^L(s)$ exists.
4) There is at most one state s represented by a vertex $v \in V$, such that $Arrive^{U,L}[v] = \emptyset$ and
there is at least one state s' represented by a vertex $v' \in V$, such that $Arrive^U[v'] \neq \emptyset$ and
there is at least one state s'' represented by a vertex $v'' \in V$, such that $Arrive^L[v''] \neq \emptyset$.
5) The input portion of each SUIO sequence is a synchronizable input sequence for each state s in M and can follow a transition entering state s without creating a synchronization problem.

The first assumption dramatically reduces the length of the state cover. Without this assumption, the resulting synchronizable checking sequence of the FSM will be restrictively long as it will have to rely on locating sequences [HE 64] which in general have exponential length. In addition, the application of a reset breaks the connection in most real protocols, and thus can be utilized to form test cases from the resulting synchronizable checking sequence. The last four assumptions are necessary for any method that will attempt to construct a synchronizable checking sequence of a given FSM using UIOs. The second assumption assures that every transition of the FSM is part of a synchronizable transition sequence starting at the initial state. The third assumption assures that for each transition of the FSM, there is an SUIO synchronizable with the transition so that one can construct the state and transition covers. The fourth assumption requires that there should be at least one state that is reached by transitions whose inputs are related to U (or L). The fifth assumption requires that any SUIO should be a synchronizable sequence for all states of the FSM. The last two assumptions assure that the uniqueness of the SUIOs in an implementation of the given FSM can be verified.
It follows from the asumptions that, $Depart^U[v] \neq \emptyset$ and $Depart^L[v] \neq \emptyset$ for each $v \in V$. Thus, there will be v^L and $v^U \in V'$ for each $v \in V$ and one can modify the procedure given in Section 3.1 by deleting the conditions for creating v^L and v^U in (1) and for creating $v^{U,L}$ in (3)c.

The proposed method utilizes $G'=(V', E')$ constructed from $G=(V, E)$ and proceeds as follows:
Step 1: Let I_{vU} and I_{vL} denote the input portions of SUIO$^U(v)$ and SUIO$^L(v)$, respectively.

Let v in $G=(V, E)$ stand for the state s in S.

a) For each vertex v^U and v^L in V',
construct I_{vU} and I_{vL} if both SUIO$^U(v)$ and SUIO$^L(v)$ must exist
(If both $Arrive^U[v]\neq\emptyset$ and $Arrive^L[v]\neq\emptyset$). Call the shorter of $\{I_{vU}, I_{vL}\}$ $I_{vU,L}$.
Otherwise, construct either I_{vU} if SUIO$^U(v)$ must exist or I_{vL} if SUIO$^L(v)$ must exist.
(The nonempty sequence I_{vU} will be used in
1) verifying a transition that is in $Arrive^U[v]\neq\emptyset$
(or if $Arrive^U[v]=\emptyset$ and SUIO$^L(v)$ does not exist, in $Arrive^{U,L}[v]$)
2) part of the verification of state s in N)
(The above statement in parantheses also holds for I_{vL}).

b) construct sets Q^U and Q^L of input sequences corresponding to input portions of SUIO sequences that will be applied to each vertex v in V for state verification:
$Q^U = \{I_{vU}$ | for each v^U that has a SUIO$^U(v)$ and I_{vU} is not a prefix of I_{wU}, $v\neq w$ in $V\}$.
$Q^L = \{I_{vL}$ | for each v^L that has a SUIO$^L(v)$ and I_{vL} is not a prefix of I_{wL}, $v\neq w$ in $V\}$.

Step 2: Assume that when a reset input r is applied, the next vertex is $v_1^{U,L}$.

Denote by T_{iU} and T_{iL} the shortest transfer sequence on $G'=(V', E')$ from the initial vertex $v_1^{U,L}$ to vertices v_i^U and v_i^L respectively. If vertex $v_i^{U,L}$ exists then denote by $T_{iU,L}$ the shortest transfer sequence on $G'=(V', E')$ from the initial vertex $v_1^{U,L}$ to vertex $v_i^{U,L}$ and let $T_{iU} = T_{iL} = T_{iU,L}$.

Let m^U denote $|Q^U|$, m^L denote $|Q^L|$, and $Q^\theta(i)$ denote the i^{th} element of Q^θ (=Q^U or Q^L). Construct an input sequence C, called *cover* of $G'=(V', E')$, that contains:

a) a state cover which is the concatenation of
$rT_{iU}\,Q^U(1)rT_{iU}\,Q^U(2)\ \ldots\ rT_{iU}Q^U(m^U)rT_{iL}\,Q^L(1)rT_{iL}\,Q^L(2)\ \ldots\ rT_{iL}Q^L(m^L)$, $\forall i$, $1 \leq i \leq n$.

b) a transition cover which consists of a *test segment* $rT_{head(e)}\,xI_{tail(e)}$
for each edge e' in E_c corresponding to the edge e in $G=(V, E)$ where $T_{head(e)}$ is the same transfer sequence on $G'=(V', E')$ from $v_1^{U,L}$ to $head(e)$ as the one used in a).

Clearly, given an appropriate choice of transfer sequences, the state cover contains a test segment for each edge that is the last edge traversed by a transfer sequence before a state is verified. Therefore, a test segment for each such edge in E_c need not be included in the transition cover.

Proposition 4: Cover C of $G'=(V', E')$ exists.

Proof:
By assumption 2, for any edge $e \in E$, there is a synchronizable path starting at the initial vertex and containing e. Thus, by construction of G', for any $e' \in E_c$, there is a path on G' starting at the initial vertex $v_1^{U,L}$ and containing e'. Every vertex in $V^U \cup V^L$ is the head of some edge in E_c and every vertex in $V^{U,L}$ is the tail of some edge in E_c. Therefore, for each vertex $v \in V'$, there is a transfer sequence on G' from vertex $v_1^{U,L}$ to vertex v and as stated in Proposition 3, v is reachable from $v_1^{U,L}$. Also, for each vertex $v \in V'$, there is an edge from v to $v_1^{U,L}$ with the label r/(-, -), i.e., a reset edge. Therefore, G' is strongly connected. Moreover, by assumptions 3 and 4; and by construction of Q^θ, Q^θ is not empty and, by assumption 5, every element of Q^θ is the input portion of the label of a path on G' starting at a vertex $v \in V'$. Therefore, all the subsequences of C of the type rI, $I \in X^*$, are defined on G', i.e., cover C of G' exists. **EOP.**

Theorem 1:
Let $G' = (V', E')$ be a digraph constructed from a given digraph $G = (V, E)$ representing an FSM M and let state cover C_1 of G' be the input portion of an IO-sequence R_1 which is the label of a path P_1 starting at vertex $v_1^{U,L}$ of G'. Then if R_1 is also an IO-sequence for implementation $N = (S', X, Y, \delta', \lambda')$ of $\Phi(M)$ then N has n states and the SUIO sequences for each state of M are also unique in N.

Proof:
First, it must be established that R_1 is a synchronizable IO-sequence for M. This is the case since, by construction, state cover C_1 is an input sequence on G'. Therefore, R_1 is a synchronizable sequence as it is the label of a path in G'. Hence, R_1 is a synchronizable IO-sequence for M.

Second, it must be shown that if R_1 is also an IO-sequence for an implementation N in $\Phi(M)$ then N has n states and the SUIO sequences for each state of M are also unique in N. In order to show this is the case, note that, any state of M is in one of the following two disjoint subsets of S:

1. $S_1 = \{s_i \in S \mid v_i^{U,L} \notin V'\}$. By assumption 4, $|S_1| \leq 1$ and if $S_1=\{s_i\}$, for some s_i, by assumption 5, v_i^U and v_i^L are in V'.
2. $S_2 = \{s_i \in S \mid v_i^{U,L} \in V'\}$. By the construction of G', $v_i^{U,L}$ is the tail of some edge in E_C and, by the construction of $Q^U \cup Q^L$, at least one of $\{I_{S_iU}, I_{S_iL}\}$ is an element of $Q^U \cup Q^L$. Therefore, the state recognition for a state $s_i \in S_2$ is achieved by $rT_{V_iU,L}\ Q^U(1)rT_{V_iU,L}\ Q^U(2) \ldots rT_{V_iU,L}\ Q^U(m^U)\ rT_{V_iU,L}\ Q^L(1)rT_{V_iU,L}\ Q^L(2) \ldots rT_{V_iU,L}\ Q^L(m^L)$.

Since r is correctly implemented in N and N is deterministic, if T is some transfer sequence, N will always be in the same state after the application of rT. Thus, since R_1 is an IO-sequence for N, the states of N reached by the transfer sequences for states in S_2 are pairwise distinguished by $Q^U \cup Q^L$. Therefore, N has at least $|S_2|$ distinct states.

To each state s of M in S_2, there corresponds a unique state s' of N, such that $\forall I \in Q^U \cup Q^L$, $\lambda'(s', I) = \lambda(s, I)$. Let $S'_2 = \{s' \mid s \in S_2\}$. If $S_1 = \emptyset$, then N has exactly n states and $\forall s' \in S'_2$, $\forall I \in Q^U \cup Q^L$, $\lambda'(s', I) = \lambda(s, I)$ and thus the result follows.

Suppose now that $S_1 \neq \emptyset$, say $S_1 = \{s_k\}$. Then $|S_2| = n$-1, so N has already n-1 determined states and both v_k^U and v_k^L exist. Then the sequence I_{S_kU} must exists and in M this distinguishes s_k from the states in S_2. Thus, since R_1 is an IO-sequence for N, the state of N reached by T_{S_kU} is not in S'_2 and so N has n states. A similar argument follows for v_k^L. Further, since the states of N reached by T_{S_kU} and T_{S_kL} are not contained in S'_2, and $|S' - S'_2|=1$, the states of N reached by T_{S_kU} and T_{S_kL} must be the same. Thus s_k is identified in N by the use of either I_{S_kL} or I_{S_kU} and so the SUIO sequences for each state of M are also unique in N as required. **EOP**

Theorem 2:
Let $G' = (V', E')$ be a digraph constructed from a given digraph $G = (V, E)$ representing an FSM M. Let transition cover C_2 of G' be the input portion of an IO-sequence R_2 which is the label of a path P_2 starting at vertex $v_1^{U,L}$ of G' and let state cover C_1 be the input portion of an IO-sequence R_1 which is the label of a path P_1 starting and ending at vertex $v_1^{U,L}$ of G'. Then the input portion of $R = R_1R_2$ is a synchronizable checking sequence of M.

Proof:
First, R is a synchronizable IO-sequence of M since, by construction, C_1C_2 is an input sequence on G'. Second, by Theorem 1, if R is also an IO-sequence for an implementation N in $\Phi(M)$ then

N has n states and the SUIO sequences for each state of M are also unique in N. Thus there is a one-to-one correspondence f from the states of M to the states of N defined by the SUIOs.

In order to complete the proof, the following must be shown:
suppose R is an IO-sequence for an implementation N in $\Phi(M)$. Let $(a, b; x/y)$ be an edge of G, then $(f(a), f(b); x/y)$ is a transition of N.

In order to show this is the case, let $e = (a, b; x/y)$ be an edge of G. Then there is a corresponding edge e' in E_C. The subsequence of R for the verification of e is of the form $(rT_{head(e)}xI_{tail(e)})/(\lambda(s_1, T_{head(e)})y\lambda(tail(e), I_{tail(e)}))$ and, by construction, $I_{tail(e)} \in Q^U \cup Q^L$. Then, since R is an IO-sequence for N, $I_{tail(e)}$ is the input portion of an SUIO of state $b'= f(b)$ of N. Hence, the state reached by N after the application of $T_{head(e)}x$ is $b' = f(b)$.

Moreover, since R_1 is an IO-sequence for N, the state reached by N after the application of $T_{head(e)}$ is $a' = f(a)$. Therefore, $e' = (a', b'; x/y)$ is a transition of N. This holds for every transition of M. The result thus follows. **EOP**

3.3 Elimination of Potential Undetectable Output Shift Faults

The cover C, defined in the previous section, is a checking sequence in so far as it distinguishes M from any faulty FSM N in $\Phi(M)$ that does not conform to M. However, under this definition M is distinguished from a faulty FSM N in $\Phi(M)$ by observing the responses of N to the checking sequence during testing. Since in the distributed test architecture the responses of N is observed by local testers, with no global clock, there may be an observability problem and output shift faults may go undetected, although the checking sequence is synchronizable.

This section considers the problem of adapting the cover C to avoid the observability problem and gives a sufficient condition under which output shift faults cannot occur in the cover C. This condition is based around the notion of the SUIOs used being *resilient* to output shift faults. This is followed by the description of a method that augments the cover C to eliminate undetectable output shift faults.

Before considering output shift faults, a number of terms will be defined. For an FSM M, let $X_\theta \subset X$ represent inputs that can be received from tester θ, Y_θ represent outputs (including -) that can be sent to θ, and $Y \subseteq Y_U \times Y_L$. Let also π_θ denote the projection function defined over an output sequence O or input-output sequence I/O such that $\pi_\theta(O)$ returns the sequence of outputs where each output $y \in Y_\theta \setminus \{-\}$ and $\pi_\theta(I/O)$ returns the sequence of inputs and outputs where each input $x \in X_\theta$ and each output $y \in Y_\theta \setminus \{-\}$. Then, an input sequence I *locally distinguishes* states s and s' of M if there is some tester θ such that $\pi_\theta(I/\lambda(s, I)) \neq \pi_\theta(I/\lambda(s', I))$. Further, an input sequence I, upon which the behaviour of M is defined, *locally distinguishes* an FSM N in $\Phi(M)$ from M if I locally distinguishes the initial states of N and M. Also, in the following, an SUIO sequence $I/\lambda(s, I)$ for a state s of M is taken as a synchronized IO-sequence with the property that for every state s' in $S \setminus \{s\}$ I locally distinguishes s and s' (i.e. there is a tester θ such that $\pi_\theta(I/\lambda(s, I) \neq \pi_\theta(I/\lambda(s', I)))$.

3.3.1 The Effects of Resilient SUIOs in Detecting Output Shift Faults

For an FSM M, let $X_\theta \subset X$ represent inputs that can be received from tester θ, Y_θ represent outputs (including -) that can be sent to θ, and $Y \subseteq Y_U \times Y_L$. Let also π_θ denote the projection function defined over an output sequence O or input-output sequence I/O such that $\pi_\theta(O)$ returns the sequence of outputs where each output $y \in Y_\theta \setminus \{-\}$ and $\pi_\theta(I/O)$ returns the sequence of inputs and outputs where each input $x \in X_\theta$ and each output $y \in Y_\theta \setminus \{-\}$. Then, an input sequence I *locally distinguishes* states s and s' of M if there is some tester θ such that $\pi_\theta(I/\lambda(s, I)) \neq \pi_\theta(I/\lambda(s', I))$. Further, an input sequence I, upon which the behaviour of M is defined, *locally distinguishes* an FSM N in $\Phi(M)$ from M if I locally distinguishes the initial states of N and M. Also, in the following, an SUIO sequence $I/\lambda(s, I)$ for a state s of M is taken as an IO-sequence with the property that for every state s' in $S \setminus \{s\}$ there is a tester θ such that $\pi_\theta(I/\lambda(s, I) \neq \pi_\theta(I/\lambda(s', I))$.

This subsection considers the case where the effectiveness of the SUIOs used in the cover C cannot be affected by output shift faults either within the SUIOs or between SUIOs and transitions immediately preceding the SUIOs. Such SUIOs will be said to be resilient to output shift faults. It will transpire that if the SUIOs used are resilient to output shift faults then the cover C has no observability problem. The following defines what it means for an SUIO to be resilient to output shift faults.

An SUIO I/O for state s is said to be *resilient to output shift faults* if for every state $s' \in S \setminus \{s\}$ there is a tester θ such that:

a) $\pi_\theta(I/\lambda(s, I)) \neq \pi_\theta(I/\lambda(s', I))$.

b) for all $z \in (Y_\theta \setminus \{-\})^*$ we have that $\pi_\theta(I/\lambda(s, I)) \neq z\pi_\theta(I/\lambda(s', I))$.

c) for all $z \in (Y_\theta \setminus \{-\})^*$ we have that $z\pi_\theta(I/\lambda(s, I)) \neq \pi_\theta(I/\lambda(s', I))$.

The first condition states that I locally distinguishes states s and s'. The second condition ensures that if I is input when M is in state s' the different behaviour is observed by θ even if extra output is produced at θ prior to the input I. It thus ensures that a backward output shift fault cannot occur between the SUIO and the transitions immediately preceding the SUIO. The third condition ensures that if I is input when M is in state s' the different behaviour is observed by θ even if some expected output fails to be produced at θ prior to the input I. It thus ensures that a forward output shift fault cannot occur between the SUIO and the transitions immediately preceding the SUIO. The following provides a sufficient condition for SUIO I/O of state s is resilient to output shift faults.

Proposition 5:

SUIO I/O for state s is resilient to output shift faults if for each $s' \neq s$ there is a tester θ such that: $I=I_1xI_2$ for some input $x \in X_\theta$ such that $\pi_\theta(\lambda(\delta(s, I_1), xI_2)) \neq \pi_\theta(\lambda(\delta(s', I_1), xI_2))$.

Informally this means that the behaviours produced by the input of I in states s and s' differ at θ after the input $x \in X_\theta$. This is a sufficient condition because differences at θ after the input x at θ cannot be involved in output shift faults with transitions before the input of x.

Suppose the SUIOs used in forming a cover C are resilient to output shift faults. Based on this, it is possible to make the following observations about C and any FSM $N \in \Phi(M)$ that is not locally distinguished from M by C.

a) N has n states and the SUIOs (locally) identify the states of N.

b) A transfer sequence for state s of M, used in forming C, reaches the corresponding state of N.

From these observations it is possible to conclude that the test segment, in the transition cover, for transition t of M executes the corresponding transition of N and then checks its final state. Further, since the SUIOs are resilient to output shift faults, any fault in this transition would locally distinguish N from M. The result below follows from these observations.

Theorem 3:
If the cover C, the synchronizable checking sequence of an FSM M, has been produced using SUIOs that are resilient to output shift faults then C is free from the observability problems.

3.3.2 Elimination of Undetectable Output Shift Faults in the Cover

This subsection will consider the problem of augmenting the cover C, in order to eliminate undetectable output shift faults, when some SUIOs used are not resilient to output shift faults. Suppose input sequence $x_1, \ldots, x_q$ is being used and the tester wished to detect output shift faults within this. It has been noted [YT 98] that this can be achieved by executing every prefix of $x_1, \ldots, x_q$. Since this may lead to a massive increase in the size of the checking sequence we will analyse the structure of C in order to limit the number of prefixes that need to be added.

Suppose the cover C does not locally distinguish an $N \in \Phi(M)$ from M. Consider the state cover. Since the role of this is to verify the transfer sequences and the uniqueness of SUIOs it is only important to determine whether output may be shifted between the input/output pairs induced by the last inputs of transfer sequences and SUIOs. Let $\tau(T)$ represent the input-output sequence induced by a transfer sequence T. Thus, $\tau(T) = T/\lambda(s_1, T)$. We may observe that $\tau(T)$ of any transfer sequence T that is followed by an SUIO^L cannot have an output shift fault, between $\tau(T)$ and SUIO^L, at port L since SUIO^L starts with input at L. Thus, in C there can be no output shift faults, involving L, between $\tau(T)$ of a transfer sequence T and any of the SUIO^{L}s. From this it follows that $\tau(T)$ must lead to the expected output at L and thus $\tau(T)$ cannot participate in an output shift fault involving L. A similar argument shows that $\tau(T)$ of any transfer sequence T that is followed by an SUIO^U cannot have an output shift fault, between $\tau(T)$ and SUIO^U, at U. Thus, $\tau(T_{iU,L})$ of any transfer sequence $T_{iU,L}$ that is separately followed by an SUIO^L and an SUIO^U cannot participate in an output shift fault.

Suppose now that there is some state i such that $T_{iU} \neq T_{iL}$. There may be at most one such state. Here there may be an output shift fault between $\tau(T_{iU})$ and the SUIO^U and between $\tau(T_{iL})$ and the SUIO^L. Thus the test sequence is augmented with rT_{iU} and rT_{iL}.

Consider now the transition cover and the sequence from this that tests transition $t=(s_i, s_j; x/y)$. Here t is tested by an input sequence in the form of a transfer sequence T followed by the input of x and then the input portion of an SUIO^U (SUIO^L) I/O to check the state reached after x. Given any transfer sequence T, the correct behaviour being observed by each tester, when applying the (possibly augmented) state cover guarantees that there can be no output shift fault between $\tau(T)$ and the following transitions. It is now sufficient to determine whether there may be an output shift fault between $x/\lambda(s_i, x)$ and $I/\lambda(s_j, I)$. Here we may note that if x is followed by I then an output shift fault between $x/\lambda(s_i, x)$ and $I/\lambda(s_j, I)$ can at most involve one output value being shifted at $L(U)$. Further, since the result of executing the SUIOs from each state has been verified during the application of the state cover, an output shift fault may only occur if an incorrect output in t compensates for SUIO^U (SUIO^L) I/O being executed from some state other than s_j. Thus an output shift fault may occur between $x/\lambda(s_i, x)$ and $I/\lambda(s_j, I)$ only if there is some state $s' \in S \backslash \{s_j\}$ such that one of the following holds.

a) Transition t involves output z at L and $z\pi_L(I/\lambda(s_j, I)) = \pi_L(I/\lambda(s', I))$. This might allow a backward output shift fault to go undetected, a fault leading to t not producing z at L being masked by the final state of t being s'.

b) Transition t involves no output at L and there exists $z \in Y_L$ such that $\pi_L(I/\lambda(s_j, I)) = z\pi_L(I/\lambda(s', I))$.This might allow a forward output shift fault to go undetected, the transition t erroneously producing output z at L and moving to state s'.

There are equivalent necessary conditions for an output shift fault to be able to occur between a transition being tested and an SUIO^L. If an output shift fault may occur between $x/\lambda(s_i, x)$ and $I/\lambda(s_j, I)$ the test is augmented with the input sequence rTx.

It is important to note that, due to their role in the cover C, it is not necessary to consider the possibility of output shift faults *within* the transfer sequences. This is because a transfer sequence T is only used to reach a state of N and not to (directly) check the input/output behaviour of any transition triggered by T. Having eliminate the possibility of output shift faults between the transfer sequences and the SUIOs in the state cover then, assuming each tester sees the expected behaviour when N is tested with C, each transfer sequence must reach the appropriate state of N (given the mapping between states of M and N defined by the SUIOs). Based on this, the transition cover then tests the individual transitions.

4 AN EXAMPLE

Consider the FSM M, represented by the digraph depicted in Figure 2, where input a is applied by upper tester U and input b is applied by lower tester L. The output 0 is sent to U and output 1 and 2 are sent to L.

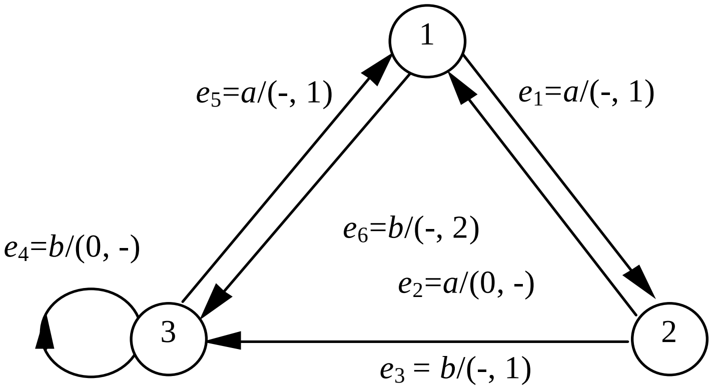

Figure 2 Digraph $G = (V, E)$.

Edge types and eligible successors are:

e_1 = (1, 2), label (e_1) = i^U/o^L, Eligible Successors: e_2, e_3

e_2 = (2, 1), label (e_2) = i^U/o^U, Eligible Successor : e_1

e_3 = (2, 3), label (e_3) = i^L/o^L, Eligible Successor : e_4

e_4 = (3, 3), label (e_4) = i^L/o^U, Eligible Successors: e_4, e_5

e_5 = (3, 1), label (e_5) = i^U/o^L, Eligible Successors: e_1, e_6

e_6 = (1, 3), label (e_6) = i^L/o^L, Eligible Successor : e_4

Arrive and Depart sets for all vertices of G are determined as in Table 2.

Table 2 Arrive and Depart sets for each vertex.

vertex	*Depart*^{U}	*Depart*^{L}	*Arrive*^{U}	*Arrive*^{L}	*Arrive*^{U,L}

1	e_1	e_6	e_2		e_5
2	e_2	e_3			e_1
3	e_5	e_4		e_3, e_6	e_4

The construction of the digraph G' proceeds as follows:
For each vertex v in V, $Depart^U[v]$ and $Depart^L[v]$ are not empty so vertices 1^U, 1^L, 2^U, 2^L, 3^U, 3^L are created in V'. Then every edge of E is considered:

- e_1 is in $Depart^U[1]$ and $Arrive^{U,L}[2]$, then construct $2^{U,L}$ in $V^{U,L}$, and create edges $(2^{U,L}, 2^U; -/(-, -))$ and $(2^{U,L}, 2^L; -/(-, -))$ in F and edge $e'_1 = (1^U, 2^{U,L}\ ; a/(-, 1))$ in E_c.
- e_2 is in $Depart^U[2]$ and $Arrive^U[1]$, then construct edge $e'_2 = (2^U, 1^U\ ; a/(0, -))$ in E_c.
- e_3 is in $Depart^L[2]$ and $Arrive^L[3]$, then construct edge $e'_3 = (2^L, 3^L\ ; b/(-, 1))$ in E_c.
- e_4 is in $Depart^L[3]$ and $Arrive^{U,L}[3]$, then construct $3^{U,L}$ in $V^{U,L}$, and create edges $(3^{U,L}, 3^U; -/(-, -))$ and $(3^{U,L}, 3^L; -/(-, -))$ in F and edge $e'_4 = (3^L, 3^{U,L}\ ; b/(0, -))$ in E_c.
- e_5 is in $Depart^U[3]$ and $Arrive^{U,L}[1]$, then construct $1^{U,L}$ in $V^{U,L}$, and create edges $(1^{U,L}, 1^U; -/(-, -))$ and $(1^{U,L}, 1^L; -/(-, -))$ in F and edge $e'_5 = (3^U, 1^{U,L}\ ; a/(-, 1))$ in E_c.
- e_6 is in $Depart^L[1]$ and $Arrive^L[3]$, then construct edge $e'_6 = (1^L, 3^L\ ; b/(-, 2))$ in E_c.

SUIO sequences for each state are:

$\text{SUIO}^U(1) = a/1, a/0$	$\text{SUIO}^L(1) = b/2$
$\text{SUIO}^U(2) = a/0$	$\text{SUIO}^L(2) = b/1$
$\text{SUIO}^U(3) = a/1, a/1$	$\text{SUIO}^L(3) = b/0$

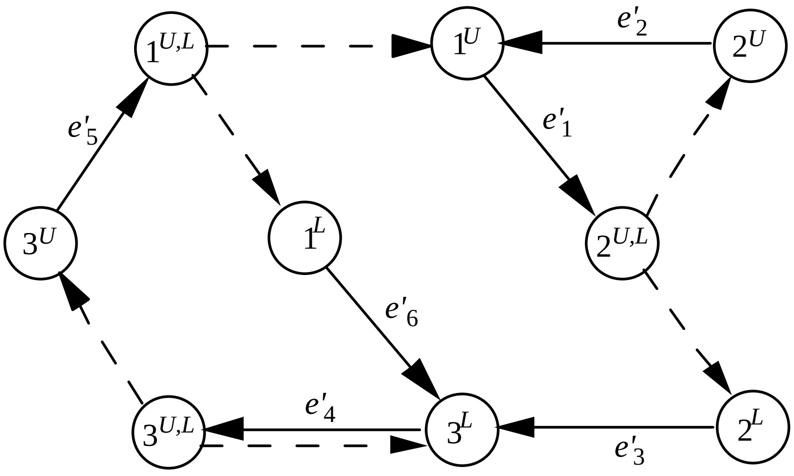

Figure 3 The Digraph G' of G in Figure 2.Edges in F are dashed.

Hence, the input portions of the SUIO sequences are:
$I_{1^U} = aa$, $I_{1^L} = b$; $I_{2^U} = a$, $I_{2^L} = b$; and $I_{3^U} = aa$, $I_{3^L} = b$. Since there exist both I_{v^U} and I_{v^L} for each vertex v in V, $I_{v^{U,L}} = I_{v^L} = b$. Finally, $Q^U = \{aa\}$, $Q^L = \{b\}$.

The construction of a cover C of G' proceeds as follows:
$T_{1^U} = T_{1^L} = T_{1^{U,L}} = \varepsilon$; $T_{2^U} = T_{2^L} = T_{2^{U,L}} = a$; and $T_{3^U} = T_{3^L} = T_{3^{U,L}} = bb$.

1) State Cover:
$rT_{1^U}\ aarT_{1^L}\ brT_{2^U}\ aarT_{2^L}\ brT_{3^U}\ aarT_{3^L}\ b = raarbraaarabrbbaarbbb.$

2) Transition Cover:
It contains a test segment for each transition except for edge e'_1 (a test segment (rab) for e'_1 is already included in the state cover since e'_1 is the last edge of $T_{2U,L}$)
i.e.,
test segment for $e'_2 = (2^U, 1^U; a/(0, -))$ is $rT_{2U}\ aI_{1U} = raaaa$
test segment for $e'_3 = (2^L, 3^L; b/(-, 1))$ is $rT_{2L}\ bI_{3L} = rabb$
test segment for $e'_4 = (3^L, 3^{U,L}; b/(0, -))$ is $rT_{3L}\ bI_{3U,L} = rbbbb$
test segment for $e'_5 = (3^U, 1^{U,L}; a/(-, 1))$ is $rT_{3U}\ aI_{1U,L} = rbbab$
test segment for $e'_6 = (1^L, 3^L; b/(-, 2))$ is $rT_{1L}\ bI_{3L} = rbb$.
Hence the transition cover is *raaaarabbrbbbbrbbabrbb.*
The state cover and the transition cover will result in the cover C of G as *raarbraaarabrbbaarbbbraaaarabbrbbbbrbbabrbb.*
C is a synchronizable checking sequence of length 43.

It is straightforward to demonstrate that the SUIOs used in C satisfy the sufficient conditions, given in Proposition 5, for SUIOs to be resilient to output shift faults. Consider, for example, the input of b at L. This forms the input of the SUIO^{L}s for each state. The expected output is: 2 at L for s_1; 1 at L for s_2; and 0 at U (and thus null at L) for s_3. For each pair of states there is a difference in the output at L after the input of b at L. By Theorem 3, since the SUIOs are resilient to output shift faults, the checking sequence is free from observability problems.

5 CONCLUSIONS

A method for constructing a checking sequence of a given FSM M using UIOs has been proposed. The resulting checking sequence does not pose controllability and observability problems during its application in a distributed test architecture. The length of the checking sequence constructed by this method may be easily reduced by eliminating redundancies and by making a wise choice of the transition sequences, and of the SUIO sequences to apply. The reduction in length can be achieved by three complementary approaches: The first approach is to eliminate rI, $I \in X^*$, in the state or transition cover if there is an rI', $I' \in X^*$ in the state or transition cover such that I is a prefix of I'. By following this approach, the cover C in our example, is reduced to *rbbaaraaaarabbrbbbbrbbab* with a length of 24 inputs by eliminating *raa, rb, raaa*, *rab*, and *rbbb* from the state cover and *rbb* from the transition cover.

In the proposed method, the same transfer sequences are used in both state and transition verification parts. By choosing different transfer sequences, the second approach of length reduction could increase the possibility of overlapping the test segments or of using shorter transfer sequences to a given state prior to verifying its outgoing edges. For instance, in the cover C in our example, the test segment (e'_4) = *rbbbb*. One could use transfer sequence $T_{3L} = b$ to form the test segment(e'_4) = *rbbb* which would reduce the length of C by 1.

In the case that both SUIO^U and SUIO^L exists for a state s, the proposed method requires that $I_{vU,L}$ should be the shorter of $\{I_{vU}, I_{vL}\}$. The third approach of length reduction is to choose the one that contributes to the greater reduction in the length of C. For instance, in the cover C in our example, for e'_4, there would be two possible test segments : *rbbbb* and *rbbbaa* and for e'_5 there also would be two possible test segments : *rbbab* or *rbbaaa*. Choosing test segment(e'_5) = *rbbaaa* implies that *rbbaa* can be deleted from C. Combining these three approaches yields C =

raaaarabbrbbbrbbaaa which is of length 19.

Naturally, when considering possible optimizations, it is import to guarantee that any changes made maintain the cover being a checking sequence that is free from observability problems. For example, when eliminating a prefix *rI* of *rI'* it is important to verify that this change cannot allow an output shift fault to go undetected. The introduction of safe optimizations will form part of future research.

Acknowledgments

This work is supported in part by the Natural Sciences and Engineering Research Council of Canada under grants STR0149338 and OGP0000976. The authors wish to thank Stephanie Guyot for her input to an earlier version of the work presented here.

REFERENCES

[AA 88] A. Aho, A.T. Dahbura, D. Lee, and M.U. Uyar, “An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours,” *Protocol Specification, Testing, and Verification*, S. Aggarwal and K. K. Sabnani, eds., North-Holland, Amsterdam, 75-86, 1988.

[BU 91] S. Boyd and H. Ural, “The synchronization problem in protocol testing and its complexity,” *Information Processing Letters*, **40**, 131-136, 1991.

[CR 99] L. Cacciari and O. Rafiq, “Controllability and observability in distributed testing,” *Information and Software Technology*, **41**, 767-780, 1999.

[CV 89] W.Y.L. Chan, S.T. Vuong, and M.R. Ito, “An improved protocol test generation procedure based on UIOS,” *Proc. SIGCOMM'89* , 283-294, 1989.

[CU 95] W. Chen and H. Ural, “Synchronizable checking sequences based on multiple UIO sequences,” *IEEE/ACM Transactions on Networking*, **3**, 152-157, 1995.

[DS 90] A.T. Dahbura, K.K. Sabnani, and M.U. Uyar, “Formal methods for generating protocol conformance test sequences,” *Proc. of IEEE,* **78**, 1317-1325, 1990.

[GI 62] A. Gill, *Introduction to the Theory of Finite-State Machine*, McGraw-Hill, New York, 1962.

[GU 95] S. Guyot and H. Ural, “Synchronizable checking sequences based on UIO sequences,” Proc. *IFIP IWPTS'95*, Evry, France, 395-407, Sept. 1995.

[HE 64] F.C. Hennie, “Fault detecting experiments for sequential circuits”, in: *Proc. Fifth Ann. Symp. Switching Circuit Theory and Logical Design* , 95-110, 1964.

[HI 97] R.M. Hierons, “Testing from a finite state machine: extending invertibility to sequences”, *The Computer Journal*, **40**, 220-230, 1997.

[II 95] ISO/IEC Open Systems Interconnection – Conformance Testing Methodolgy and Framework, 9646-1, 1995.

[LY 94] D. Lee and M. Yannakakis, “Testing finite state machines: state identification and verification,” *IEEE Transactions on Computers*, **43**, 306-320, 1994.

[LB 94] G. Luo, R. Dssouli, G. v. Bochmann, P. Venkataram and A. Ghedamsi, “Test generation with respect to distributed interfaces,” *Computer Standards and Interfaces*, **16**, 119-132, 1994.

[SD 88] K.K. Sabnani and A.T. Dahbura, “A protocol test generation procedure,” *Computer Networks*, **15**, 285-297, 1988.

[SB 84] B. Sarikaya and G. v. Bochmann, “Synchronization and specification issues in protocol testing,” *IEEE Transactions on Communications*, **32**, 389-395, 1984.

[TY 98] K.C. Tai and Y.C. Young, “Synchronizable test sequences of finite state machines,” *Computer Networks*, **13**, 1111-1134, 1998.

[UW 93] H. Ural and Z. Wang, “Synchronizable test sequence generation using UIO sequences,” *Computer Communications*, **16**, 653-661, 1993.

[YT 98] Y.C. Young and K.C. Tai, “Observation inaccuracy in conformance testing with multiple testers,” Proc. *IEEE WASET*, 80-85, 1998.